Referências Bibliográficas

Asirvatham, A. & Hoppe, H., 2005. Terrain Rendering Using GPU-Based Geometry Clipmaps. In *GPU Gems 2*. NVidia Corporation. pp.27-45.

Boyce, W.E. & DiPrima, R.C., 2001. *Elementary Differential Equations and Boundary Value Problems*. 7th ed. John Wiley & Sons, Inc.

Chen, J. & Lobo, N., 1995. Toward interactive-rate simulation of fluids with moving obstacles using navier-stokes equations. In *Graphical models and Image Processing*., 1995.

Cormen, T.H., Leiserson, C.E., Rivest, R.L. & Stein, C., 2001. *Introduction to Algorithms*. 2nd ed. Boston: Elsevier.

Finch, M., 2004. Effective Water Simulation from Physical Models. In N. Corporation, ed. *GPU Gems*. Addison-Wesley. p.1.

Foley, J.D., Dam, A.v., Feiner, S.K. & Hughes, J.F., 1995. *Computer Graphics: Principles and Pratice*. 2nd ed. Addison-Wesley.

Foster, N. & Metaxas, D., 2000. Modeling water for computer animation. *Commun ACM*, 43(7), p.60–67.

Fournier, A. & Reeves, W.T., 1986. A simple model of ocean waves. *ACM SIGGRAPH Computer Graphics*, 20(4), pp.75 - 84.

Frigo, M. & Johnson, S.G., 2005. The Design and Implementation of FFTW3. *Proceedings of the IEEE*, 93(2), pp.216-31.

Gamma, E., Helm, R., Johnson, R. & Vlissides, J., 1995. *Design Patterns: Elements of Reusable Object-Oriented Software*. 1st ed. Upper Saddle River, New Jersey, USA: Addison-Wesley Professional.

Gerstner, F.J., 1809. Theorie der wellen. *Ann. der Physik 32*, pp.412-40.

González, X.X.V., 2009. *Hydrax - Ogre Wiki*. [Online] Available at: http://www.ogre3d.org/wiki/index.php/Hydrax [Accessed 10 November 2009].

Guardado, J. & Sánchez-Crespo, D., 2004. Rendering Water Caustics. In Fernando, R. *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics*. Addison-Wesley Professional. pp.31-44.

Hecht, E., 1987. *Optics*. 2nd ed. Addison Wesley.

Heidrich, W. & Seidel, H.-p., 1999. Realistic, hardware-accelerated shading and lighting. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

Hu, Y. et al., 2004. Realistic, Real-Time Rendering of Ocean Waves. *Computer Animation and Virtual Worlds*, 17(1), pp.59 - 67.

Imamiya, A. & Zhang, D., 1995. Modelling breaking ocean waves, influence of floor. In *Pacific Graphics 95*. Seoul, Korea, 1995.

Jensen, L.S. & Golias, R., 2001. *Deep-Water Animation and Rendering*. [Online] (1.0) Available at: http://www.gamasutra.com/gdce/2001/jensen/jensen_01.htm [Accessed 23 Novembro 2009].

Johanson, C., 2004. *Real-time water rendering*. MSc thesis. Lund University.

Kass, M. & Miller, G., 1990. Rapid, stable fluid dynamics for computer graphics. In *17th annual conference on Computer graphics and interactive techniques*. New York, 1990. ACM Press.

Kinsman, B., 1965. *Wind Waves: Their Generation and Propagation on the Ocean Surface*. Dover Pubns.

Kozlov, S., 2004. Perspective Shadow Maps: Care and Feeding. In N. Corporation, ed. *GPU Gems*. Addison-Wesley.

Kryachko, Y., 2005. Using Vertex Texture Displacement for Realistic Water Rendering. In *GPU Gems 2*. NVidia Corporation. pp.283-94.

Mastin, G.A., Watterger, P.A. & Mareda, J.F., 1987. Fourier Sysnthesis of Ocean Scenes. *IEEE Computer Graphics and Applications*, 7(3), pp.16-23.

McCrae, J., Mordatch, I., Glueck, M. & Khan, A., 2009. Multiscale 3D navigation. In Spencer, S.N., ed. *Proceedings of the 2009 symposium on Interactive 3D graphics and games*. Boston, Massachusetts, USA, 2009. ACM Press.

McDowell, P.L., 2009. The Delta3D Game Engine is Reaching Maturity. *MS&T Magazine*, I(5), pp.22-24.

Mitchell, J.L., 2005. *Real-Time Synthesis and Rendering of Ocean Water*. Technical Report. ATI Research.

Nilson, J.W. & Riedel, S.A., 2001. *Eletric Circuits*. 6th ed. Prentice-Hall.

NVIDIA Corporation , 2009. *NVIDIA CUDA™: Programming Guide*. [Online] NVIDIA Corporation (2.3.1) Available at:

http://developer.download.nvidia.com/compute/cuda/2_3/toolkit/docs/NVIDIA_CUDA_Programming_Guide_2.3.pdf [Accessed 20 Setembro 2009].

NVidia Corporation, 2009. *CUDA Zone*. [Online] NVidia Corporation (2.3) Available at: http://developer.download.nvidia.com/compute/cuda/2_3/toolkit/docs/NVIDIA_CUDA_BestPracticesGuide_2.3.pdf [Accessed 25 Setembro 2009].

Peachey, D.R., 1986. Modeling waves and surf. *ACM SIGGRAPH Computer Graphics*, 20(4), pp.65 - 74.

Perlin, K., 1985. An Image Synthesizer. *Computer Graphics*, 19(3), pp.287-96.

Perlin, K., 2002. Improving noise. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. San Antonio, Texas, 2002. ACM Press.

Phong, B.T., 1975. Illumination for computer generated pictures. *Communications of the ACM*, 8(6), pp.311-17.

Pozzer, C.T., 2000. *Uso de Técnicas de Síntese de Imagem Aplicadas a um Ambiente de Representação de Superfícies Líquidas Estáticas e Dinâmicas*. Msc Thesis. São José dos Campos: ITA.

Salgado, A., Conci, A. & Clua, E., 2007. Simulação Visual de Ondas Oceânicas em Tempo Real Usando a GPU. In Walter, M. & Silveira, L.G.J., eds. *Proceedings of VI Brazilian Symposium on Computer Games and Digital Entertainment*. São Leopoldo - RS, 2007. Sociedade Brasileira de Computação - SBC.

Santos, P.I.N. & Celes, W., 2009. *Ray Tracing Dynamic Scenes on the GPU*. Msc Thesis. Rio de Janeiro: PUC-Rio.

Schlick, C., 1994. An Inexpensive BRDF Model for Physically-based Rendering. *Computer Graphics Forum*, 13(3), pp.233 - 246.

Schmidt, D., Stal, M. & Rohnert, H., 2000. *Pattern-Oriented Software Architecture - Volume 2*. 1st ed. John Wiley & Sons Inc.

Sekulic, D., 2004. Efficient Occlusion Culling. In R. Fernando, ed. *GPU Gems*. 1st ed. Upper Saddle River, New Jersey, USA: Addison-Wesley.

Singh, I., 2002. *Designing Enterprise Applications with J2EETM Platform*. 1st ed. Addison-Wesley.

Sissom, L.E. & Pitts, D.R., 2001. *Fenômenos de Transporte*. 1st ed. Rio de Janeiro: LTC.

Starm, J., 1996. Random Caustics: Natural Textures and Wave Theory Revisited. In *SIGGRAPH '96: ACM SIGGRAPH 96 Visual Proceedings: The art and interdisciplinary programs of SIGGRAPH '96*. New Orleans, Louisiana, USA: ACM Press. p.150.

Su, K.L., 1996. *Analog Filters*. 1st ed. New York - NY - USA: Chapman & Hall.

Sýkora, D. & Jelínek, J., 2002. *Efficient View Frustum Culling*. Technical Report. Prague - Czech Republic: Czech Technical University.

Tessendorf, J., 1999. Simulating Ocean Water. In *SIGGRAPH'99 Course Notes*., 1999.

Ts'o, P.Y. & Barsky, B.A., 1987. Modeling and rendering waves: wave-tracing using beta-splines and reflective and refractive texture mapping. *ACM Transactions on Graphics*, 6(3), pp.191 - 214.

WikiWaves, 2008. *Ocean-Wave Spectra*. [Online] Available at: http://www.wikiwaves.org/index.php/Ocean-Wave_Spectra [Accessed 10 Novembro 2009].

Anexo 1 – Tabelas de Teste

A organização dos dados segue o seguinte padrão:

Média (Devio-Padrão [Intervalo de Confiança])

Observa-se que o nível de significância utilizado é de 5%.

Tabela 23: Resultados da Variação do Número de Harmônicas de Gerstner [Bruto]

	CPU 1	CPU 2	CPU 4	CUDA	Shader
1	25.658 (0.83642 [26.942398-23.374088])	11.929427 (0.615374 [11.946656-11.912199])	6.325711 (0.955012 [6.345387-6.306035])	3.394855 (0.434907 [3.401682-3.388029])	1.185235 (0.350298 [1.189014-1.181457])
2	44.843985 (0.378887 [44.864370-44.823599])	22.462714 (0.335112 [22.475519-22.449909])	11.863806 (1.522620 [11.906265-11.821346])	3.552689 (0.529878 [3.561175-3.544204])	1.185130 (0.365221 [1.189072-1.181189])
3	65.719782 (0.513986 [65.753213-65.686350])	33.436407 (0.403859 [33.455190-33.417625])	17.269592 (1.685961 [17.326187-17.212996])	3.772364 (0.613279 [3.782457-3.762270])	1.192775 (0.388731 [1.196979-1.188572])
4	86.603419 (0.567284 [86.645746-86.561091])	44.027770 (0.422991 [44.050317-44.005223])	22.916911 (2.873438 [23.027787-22.806034])	4.058265 (0.462617 [4.066136-4.050395])	2.684420 (0.956665 [2.698330-2.670511])
5	107.735842 (0.566544 [107.783062-107.688623])	54.904864 (1.045430 [54.967040-54.842687])	28.583471 (3.369834 [28.728394-28.438548])	4.376332 (0.446405 [4.384187-4.368478])	3.101698 (0.731814 [3.112935-3.090461])
6	128.668406 (0.675246 [128.729912-128.606900])	64.661094 (0.795266 [64.712427-64.609762])	33.836753 (4.081393 [34.027702-33.645804])	4.749463 (0.492653 [4.758467-4.740458])	3.573705 (0.789242 [3.586564-3.560845])
7	149.836128 (0.719758 [149.906840-149.765416])	75.176980 (0.513642 [75.212707-75.141253])	39.372545 (4.795125 [39.614402-39.130688])	5.074239 (0.496072 [5.083582-5.064896])	3.892898 (0.704278 [3.904891-3.880906])
8	180.717903 (0.941137 [180.819291-180.616516])	86.608252 (0.547750 [86.649122-86.567382])	44.814944 (4.949154 [45.081026-44.548861])	5.459934 (0.455899 [5.468818-5.451051])	4.272595 (0.808369 [4.287023-4.258167])
9	191.936460 (0.789310 [192.024183-191.848736])	96.248296 (0.770962 [96.308933-96.187660])	50.335073 (5.579865 [50.653037-50.017108])	5.772859 (0.463067 [5.782116-5.763603])	4.757069 (0.877111 [4.773401-4.740736])
10	212.680182 (0.857208 [212.780408-212.579956])	106.736389 (0.623470 [106.788119-106.684659])	55.908386 (6.406578 [56.292973-55.523798])	6.149954 (0.471862 [6.159672-6.140237])	5.135881 (0.860425 [5.152485-5.119278])
11	233.901365 (0.862292 [234.007200-233.795529])	117.342909 (1.251550 [117.451743-117.234076])	62.454252 (6.166154 [62.845327-62.063178])	6.483217 (0.492787 [6.493623-6.472810])	5.603995 (0.873337 [5.621462-5.586528])
12	254.681706 (0.882159 [254.794493-254.568919])	131.892812 (1.158719 [131.999633-131.785991])	66.675817 (6.738151 [67.117262-66.234372])	6.849235 (0.478571 [6.859606-6.838863])	6.007883 (0.841230 [6.025249-5.990518])
13	275.896502 (0.913049 [276.017984-275.775020])	138.689042 (3.286273 [138.999653-138.378431])	71.964671 (6.748200 [72.424036-71.505306])	7.204217 (0.492598 [7.215144-7.193289])	6.463170 (0.862251 [6.481570-6.444770])
14	296.577792 (0.940630 [296.707507-296.448077])	152.398053 (3.587062 [152.753147-152.042960])	78.621603 (3.847576 [78.895147-78.348058])	7.580027 (0.579002 [7.593190-7.566863])	6.932321 (0.859294 [6.951214-6.913427])
1	317.940197	159.840762	82.954060	7.835066 (0.532954	7.369424 (0.910427

5	(1.008198 [318.084313-317.796080])	(3.193118 [160.164375-159.517149])	(7.085767 [83.471628-82.436491])	[7.847369-7.822762])	[7.390007-7.348841])
16	339.217113 (1.172267 [339.389811-339.044415])	174.960252 (4.373286 [175.424423-174.496081])	88.427352 (8.033649 [89.032953-87.821751])	8.202237 (0.456750 [8.213014-8.191459])	7.510239 (0.784954 [7.528367-7.492112])
17	359.803263 (1.172677 [359.981119-359.625408])	180.999230 (4.154365 [181.446776-180.551683])	93.979069 (8.179937 [94.614793-93.343345])	8.541425 (0.497860 [8.553399-8.529451])	8.218650 (0.851068 [8.238924-8.198375])
18	380.876116 (1.116148 [381.050706-380.701526])	196.670460 (4.386606 [197.163565-196.177356])	99.334822 (7.860061 [99.963222-98.706422])	8.917487 (0.482786 [8.929337-8.905636])	12.158405 (0.912260 [12.184522-12.132288])
19	401.748966 (1.155361 [401.934478-401.563454])	201.996284 (4.126910 [202.467219-201.525349])	104.873769 (9.513282 [105.656813-104.090725])	9.232548 (0.483952 [9.244628-9.220467])	9.128932 (0.877674 [9.150854-9.107009])
20	422.811035 (1.327102 [423.029311-422.592758])	212.678039 (4.649245 [213.221635-212.134443])	110.524707 (10.768182 [111.433773-109.615641])	9.582701 (0.446822 [9.594056-9.571346])	9.359300 (0.810125 [9.379768-9.338833])

Tabela 24: Resultados da Simulação na CPU [Bruto]

	128	**256**	**512**	**1024**	**2048**
g-h–CPU-1 core	17.681537 (0.228021 [17.695080-17.667994])	70.556142 (0.470171 [70.611115-70.501169])	282.359788 (1.992276 [282.823200-281.896375])	1132.366929 (13.993802 [1139.296542-1125.437316])	4557.846165 (53.297417 [4619.116786-4496.575544])
g-n–CPU-1 core	37.051982 (0.392084 [37.075269-37.028695])	142.150082 (0.713640 [142.233522-142.066642])	568.728960 (1.657210 [569.114434-568.343486])	2278.131511 (12.166684 [2284.156351-2272.106672])	9140.468597 (64.328006 [9214.419965-9066.517230])
g-h–CPU-2 core	8.916843 (0.158390 [8.923442-8.910244])	35.536345 (0.335999 [35.564248-35.508441])	141.419878 (0.676236 [141.531497-141.308260])	565.789945 (3.752240 [567.015651-564.564238])	2271.970219 (25.165874 [2290.946630-2252.993808])
g-n–CPU-2 core	17.874167 (0.272393 [17.885516-17.862818])	71.297113 (0.881886 [71.370350-71.223876])	284.846794 (1.143692 [285.035570-284.658018])	1139.385250 (4.530715 [1140.865253-1137.905247])	4566.922612 (30.110810 [4589.627768-4544.217455])
g-h–CPU-4 core	4.684415 (0.485927 [4.699114-4.669716])	19.486458 (1.148773 [19.555811-19.417106])	73.207419 (4.754482 [73.772442-72.642396])	289.496652 (6.092377 [290.934158-288.059147])	1233.289179 (32.151137 [1249.741125-1216.837233])
g-n–CPU-4 core	9.329243 (0.854906 [9.355104-9.303382])	37.075540 (2.734710 [37.240636-36.910443])	146.122963 (4.844218 [146.698650-145.547276])	584.344000 (18.822786 [588.785264-579.902736])	2341.324806 (34.403506 [2358.929304-2323.720308])
f-h–CPU-1 core	11.162011 (0.289963 [11.171983-11.152039])	44.882749 (0.639430 [44.927480-44.838018])	180.676233 (1.921707 [180.950203-180.402262])	743.6879333 (1.48217 [743.669584-743.638462])	3170.145583 (44.664004 [3201.615828-3138.675338])
f-n–CPU-1 core	6.927963 (0.221460 [6.935579-6.920347])	31.040347 (0.594773 [31.081953-30.998740])	135.683519 (1.778595 [135.937086-135.429952])	612.01872 (2.597561 [612.028673-611.976945])	2861.218381 (79.256246 [2917.062311-2805.374451])
f-h–CPU-2 core	8.588750 (0.307115 [8.598491-8.579010])	35.212993 (0.495333 [35.245301-35.180686])	143.042670 (2.342134 [143.353577-142.731763])	594.509822 (6.023733 [596.163032-592.856613])	2519.147674 (9.601185 [2525.186515-2513.108834])
f-n–CPU-2 core	6.773045 (0.342650 [6.783913-6.762178])	30.685205 (0.558203 [30.721613-30.648797])	131.150945 (1.613663 [131.365151-130.936739])	585.050650 (3.542695 [586.022940-584.078359])	2780.461510 (45.592611 [2809.137816-2751.785204])
f-h–CPU-4 core	7.684937 (0.334404 [7.695284-7.674591])	30.843460 (0.883171 [30.898867-30.788053])	126.599429 (2.402371 [126.907896-126.290961])	527.193945 (7.176526 [529.090564-525.297326])	2252.764463 (9.795114 [2258.925279-2246.603647])
f-n–CPU-4 core	6.871119 (0.220771 [6.877950-	30.102482 (0.598123 [30.140006-	129.631339 (1.107536 [129.773548-	579.810036 (3.346658 [580.694494-	2771.729549 (38.724822 [2796.086233-

	6.864289])	30.064957])	129.489131])	578.925578])	2747.372864])
p-h–CPU-1 core	1.337193 (0.165046 [1.341328-1.333058])	4.172564 (0.359639 [4.189880-4.155248])	16.369149 (1.628104 [16.524486-16.213813])	66.223444 (7.651653 [67.666525-64.780364])	344.216641 (39.409119 [360.449573-327.983710])
p-n–CPU-1 core	8.190466 (0.195954 [8.195375-8.185557])	31.660444 (0.431182 [31.681205-31.639683])	125.270031 (1.251216 [125.389408-125.150653])	493.839705 (7.398518 [495.235045-492.444364])	2077.143641 (27.509490 [2088.475020-2065.812261])
p-h–CPU-2 core	0.859538 (0.223306 [0.863641-0.855435])	2.532675 (0.196095 [2.539534-2.525816])	9.336709 (0.671696 [9.383023-9.290395])	36.831293 (2.252585 [37.139648-36.522937])	221.365244 (9.827701 [224.236639-218.493849])
p-n–CPU-2 core	4.115115 (0.181076 [4.118442-4.111788])	16.235934 (0.355436 [16.248366-16.223502])	64.422650 (0.730484 [64.473017-64.372282])	256.088156 (2.629451 [256.448101-255.728212])	1117.288825 (10.669519 [1120.406177-1114.171474])
p-h–CPU-4 core	0.628796 (0.213047 [0.631964-0.625628])	1.676272 (0.252763 [1.683202-1.669342])	5.939019 (0.763159 [5.980080-5.897958])	22.599330 (1.832167 [22.793508-22.405153])	163.267485 (8.987149 [165.372815-161.162156])
p-n–CPU-4 core	2.523919 (0.302345 [2.528415-2.519423])	9.728042 (0.862988 [9.751703-9.704380])	38.843053 (1.949858 [38.947962-38.738143])	152.426976 (5.295535 [152.988210-151.865742])	685.652001 (26.494374 [691.858574-679.445429])

Tabela 25: Resultados da Simulação na GPU [Bruto]

	128	**256**	**512**	**1024**	**2048**
g-h–GPU-CUDA	1.344180 (0.298235 [1.348175-1.340186])	2.881002 (0.375261 [2.888725-2.873278])	8.500624 (0.391538 [8.515006-8.486242])	28.400337 (0.411221 [28.428904-28.371770])	108.479937 (0.324987 [108.524316-108.435558])
g-n–GPU-CUDA	1.017835 (0.183824 [1.020296-1.015373])	3.266284 (0.338360 [3.273247-3.259320])	12.033487 (0.470452 [12.050768-12.016206])	46.454152 (0.432735 [46.484214-46.424090])	182.245073 (0.541047 [182.318957-182.171190])
g-h–GPU-Shader	0.236983 (0.217048 [0.238765-0.235201])	0.460248 (0.371287 [0.467413-0.453084])	1.715002 (0.467468 [1.723171-1.706832])	5.913951 (1.211118 [5.949954-5.877948])	23.084539 (16.015018 [23.986531-22.182547])
g-n–GPU-Shader	0.320790 (0.214493 [0.322551-0.319029])	0.4675396 (0.724326 [0.4689373-0.4661420])	1.909886 (0.543484 [1.919384-1.900389])	6.729065 (2.310416 [6.797747-6.660383])	25.044357 (9.656101 [25.588204-24.500509])
f-h–GPU-CUDA	1.458394 (0.362871 [1.463124-1.453664])	3.118333 (0.463406 [3.127173-3.109493])	9.890930 (0.471631 [9.906939-9.874921])	33.354024 (0.506530 [33.385983-33.322065])	165.830431 (16.822260 [168.073820-163.587041])
f-n–GPU-CUDA	0.774281 (0.231483 [0.777298-0.771264])	2.115308 (0.346053 [2.121909-2.108707])	7.610890 (0.477947 [7.627113-7.594666])	28.295635 (0.504334 [28.327455-28.263815])	112.307199 (11.074072 [113.784019-110.830379])
p-h–GPU-Shader	0.234018 (0.214782 [0.235783-0.232253])	0.570992 (0.239239 [0.573578-0.568405])	1.718398 (0.455999 [1.726212-1.710584])	5.913276 (0.749451 [5.934863-5.891688])	22.047738 (0.352636 [22.066704-22.028772])
p-n–GPU-Shader	0.327293 (0.208914 [0.329010-0.325576])	0.646716 (0.296068 [0.649917-0.643515])	1.769983 (0.452886 [1.777744-1.762223])	5.952674 (0.733044 [5.973789-5.931559])	22.022794 (0.647262 [22.057606-21.987982])
p-h–GPU-CUDA	0.758937 (0.654942 [0.767559-0.750316])	1.872370 (1.455787 [1.901058-1.843682])	6.224797 (4.596561 [6.385802-6.063792])	22.279615 (16.248451 [23.346507-21.212723])	87.217843 (63.193918 [95.384768-79.050918])
p-n–GPU-CUDA	0.762510 (0.659827 [0.771196-0.753825])	1.870884 (1.453168 [1.899521-1.842248])	6.215189 (4.583963 [6.375752-6.054625])	22.273043 (16.245799 [23.339760-21.206325])	86.597235 (63.311412 [94.779345-78.415125])

Tabela 26: Resultados LoD [Bruto]

	256	**512**	**1024**	**2048**

GeoClipMap	7.821250 (2.907571 [8.150266-7.492234])	11.242348 (1.066956 [11.363083-11.121613])	27.521696 (0.883279 [27.621646-27.421746])	67.153427 (2.648465 [67.453123-66.853731])
Radial LoD	6.256192 (0.874556 [6.355155-6.157229])	8.139195 (0.774768 [8.226867-8.051524])	14.083361 (0.929869 [14.188583-13.978138])	47.611446 (0.848671 [47.707481-47.515412])
Projected Grid	7.489686 (0.735025 [7.572860-7.406512])	10.024673 (1.073091 [10.146103-9.903244])	15.469318 (1.343152 [15.621307-15.317330])	49.048771 (0.797553 [49.139021-48.958521])

Tabela 27: Resultados Totais [Bruto]

	256	512	1024	2048
GeoClipMap& g-Shader	10.084365 (2.729790 [10.393263-9.775466])	15.374087 (1.649948 [15.560793-15.187382])	42.589809 (3.367859 [42.970911-42.208708])	121.274048 (1.094782 [121.397932-121.150164])
GeoClipMap& f-CUDA	15.519661 (2.199018 [15.768498-15.270824])	31.685032 (3.184657 [32.045402-31.324661])	92.224190 (1.987435 [92.449085-91.999295])	345.688995 (2.833822 [346.009666-345.368325])
GeoClipMap& p-Shader	9.618784 (2.296378 [9.878638-9.358930])	15.935342 (3.081110 [16.283995-15.586689])	40.709972 (2.967322 [41.045750-40.374195])	113.472061 (2.708025 [113.778497-113.165626])
Radial LoD& g-Shader	9.446818 (0.775702 [9.534595-9.359041])	13.161928 (0.889139 [13.262542-13.061315])	27.246414 (1.137108 [27.375087-27.117741])	100.623360 (0.512372 [100.681339-100.565381])
Radial LoD& f-CUDA	12.490114 (1.290240 [12.636116-12.344113])	28.645395 (1.638211 [28.830773-28.460018])	81.168724 (1.194770 [81.303922-81.033526])	331.802826 (1.108439 [331.928255-331.677397])
Radial LoD& p-Shader	7.767086 (0.830028 [7.861011-7.673161])	13.170369 (0.355554 [13.210603-13.130135])	29.749798 (0.494822 [29.805791-29.693805])	96.454262 (1.016561 [96.569294-96.339229])
Projected Grid & g-Shader	9.555145 (2.153959 [9.798884-9.311407])	13.749747 (0.748797 [13.834480-13.665015])	33.438900 (1.728683 [33.634515-33.243285])	98.227448 (1.077025 [98.349322-98.105573])
Projected Grid & f-CUDA	13.001146 (2.088128 [13.237435-12.764857])	29.133961 (1.627560 [29.318133-28.949789])	79.980492 (0.976744 [80.091018-79.869965])	333.887573 (2.149922 [334.130855-333.644292])
Projected Grid & p-Shader	11.020406 (0.897593 [11.121976-10.918836])	17.229994 (1.765024 [17.429721-17.030267])	33.486652 (0.424726 [33.534714-33.438591])	98.285141 (1.411577 [98.444873-98.125409])

Tabela 28: Resultados de Gerstenr com a mudança de memória *CUDA* [Bruto]

	Global	Constante	Local - Global	Local - Constante
1	8.290469 (0.466543 [8.301522-8.279416])	3.394855 (0.434907 [3.401682-3.388029])	4.689504 (0.493625 [4.698471-4.680536])	4.058514 (0.467991 [4.066476-4.050552])
2	13.080884 (0.446265 [13.094041-13.067726])	3.552689 (0.529878 [3.561175-3.544204])	5.971162 (0.503218 [5.981390-5.960933])	4.649698 (0.491247 [4.658592-4.640803])
3	17.675455 (0.462188 [17.691232-17.659679])	3.772364 (0.613279 [3.782457-3.762270])	7.331683 (0.511429 [7.343121-7.320245])	5.315550 (0.458700 [5.324376-5.306725])
4	22.295687 (0.678301 [22.321615-22.269758])	4.058265 (0.462617 [4.066136-4.050395])	8.525364 (0.482445 [8.536960-8.513769])	6.008924 (0.505575 [6.019226-5.998623])
5	27.150388 (1.337131 [27.206699-27.094077])	4.376332 (0.446405 [4.384187-4.368478])	9.825184 (0.498431 [9.837996-9.812372])	6.686762 (0.480875 [6.697068-6.676456])
6	31.533194 (0.495542 [31.555647-31.510740])	4.749463 (0.492653 [4.758467-4.740458])	11.089432 (0.487241 [11.102704-11.076159])	7.395586 (0.549407 [7.407924-7.383248])
7	36.116839 (0.606010 [36.146205-36.087474])	5.074239 (0.496072 [5.083582-5.064896])	12.271925 (0.463452 [12.285187-12.258662])	7.909454 (0.401088 [7.918757-7.900151])
8	40.731492 (0.586010 [40.761623-40.701361])	5.459934 (0.455899 [5.468818-5.451051])	13.487265 (0.463466 [13.501146-13.473385])	8.663837 (0.503050 [8.676024-8.651651])
9	45.349594 (0.509298 [45.377205-45.321983])	5.772859 (0.463067 [5.782116-5.763603])	14.886845 (0.447034 [14.900886-14.872804])	9.237342 (0.496772 [9.249742-9.224941])
10	46.554124 (0.538946	6.149954 (0.471862	16.141838 (0.512130	9.866357 (0.491396

	[46.583718-46.524530])	[6.159672-6.140237])	[16.158574-16.125101])	[9.879024-9.853690])
11	50.435349 (0.587586 [50.468918-50.401781])	6.483217 (0.492787 [6.493623-6.472810])	17.360278 (0.452085 [17.375583-17.344974])	10.530158 (0.512702 [10.543791-10.516525])
12	54.905508 (0.514420 [54.936160-54.874857])	6.849235 (0.478571 [6.859606-6.838863])	18.606840 (0.528765 [18.625352-18.588327])	11.152851 (0.487103 [11.166157-11.139545])
13	59.426399 (0.540236 [59.459883-59.392916])	7.204217 (0.492598 [7.215144-7.193289])	19.854664 (0.580764 [19.875657-19.833671])	11.813115 (0.521255 [11.827756-11.798473])
14	63.904956 (0.596595 [63.943278-63.866634])	7.580027 (0.579002 [7.593190-7.566863])	21.110148 (0.655786 [21.134569-21.085727])	12.379225 (0.466850 [12.392642-12.365808])
15	68.384416 (0.580018 [68.422958-68.345875])	7.835066 (0.532954 [7.847369-7.822762])	22.336132 (0.683498 [22.362293-22.309970])	13.001672 (0.470118 [13.015502-12.987843])
16	72.884361 (0.579302 [72.924084-72.844638])	8.202237 (0.456750 [8.213014-8.191459])	23.571292 (0.724011 [23.599747-23.542838])	13.642924 (0.454056 [13.656599-13.629249])
17	77.436460 (0.702172 [77.486088-77.386832])	8.541425 (0.497860 [8.553399-8.529451])	24.835003 (0.691957 [24.862902-24.807103])	14.360856 (0.455496 [14.374923-14.346790])
18	81.894515 (0.652473 [81.941911-81.847119])	8.917487 (0.482786 [8.929337-8.905636])	26.080513 (0.674703 [26.108379-26.052647])	15.030250 (0.459907 [15.044764-15.015736])
19	86.959966 (0.597829 [87.004703-86.915230])	9.232548 (0.483952 [9.244628-9.220467])	27.260309 (0.647408 [27.287630-27.232988])	15.741031 (0.490759 [15.756870-15.725193])
20	90.789839 (0.709040 [90.844056-90.735622])	9.582701 (0.446822 [9.594056-9.571346])	28.488108 (0.716809 [28.519017-28.457199])	16.138535 (0.519879 [16.155524-16.121545])

Anexo 2 – Código do *Shader* de Iluminação

Tabela 29: Código do Shader de Iluminação (linguagem HLSL)

```
001 float4 tex2Dbilinear( sampler2D texSam, float2 uv, float texelSize  )
002 {
003     float4 height00 = tex2D(texSam, uv);
004     float4 height10 = tex2D(texSam, uv + float2(texelSize, 0));
005     float4 height01 = tex2D(texSam, uv + float2(0, texelSize));
006     float4 height11 = tex2D(texSam, uv + float2(texelSize , texelSize));
007
008     vec2 f = fract( uv.xy * 1.0 / texelSize );
009
010     float4 tA = lerp( height00, height10, f.x );
011     float4 tB = lerp(height01, height11, f.x );
012
013     return lerp( tA, tB, f.y );
014 }
015
016
017 void main_vp(
018         float4 pos      : POSITION,
019         float4 normal       : NORMAL,
020         float2 tex      : TEXCOORD0,
021         out float4 oPos     : POSITION,
022         out float2 noiseCoord   : TEXCOORD0,
023         out float4 projectionCoord : TEXCOORD1,
024         out float3 oEyeDir : TEXCOORD2,
025         out float3 oWorldPos : TEXCOORD3,
026         uniform float4x4 worldViewProjMatrix,
027         uniform float4x4 worldMatrix,
028         uniform float3 eyePosition, // object space
029         uniform float scale,
030         uniform float texelSize,
031         uniform float far,
032         uniform float waveHeight,
033         uniform float choppyLamda,
034         uniform sampler2D diplacementMap : register(s0)
035         )
036 {
037
038     pos.xz = pos.xz + eyePosition.xz;
039     float2 DispCoord = pos.xz / scale;
040
041     noiseCoord = pos.xz / (100.0 * far);
042
043     float4 disp = tex2Dbilinear(diplacementMap, DispCoord, texelSize);
044
045     pos.xz = pos.xz + choppyLamda * dsp.xz;
046     pos.y = waveHeight * disp.y;
047
048     oPos = mul(worldViewProjMatrix, pos);
049
050     // Projective texture coordinates, adjust for mapping
051     float4x4 scalemat = float4x4(0.5,   0,   0, 0.5,
052                                  0,-0.5   0, 0.5,
053                        0,   0, 0.5, 0.5,
054                                     0,   0,   0,   1);
055
056     projectionCoord = mul(scalemat, oPos);
057
058     oWorldPos = mul(worldMatrix,pos).xyz;
059
060    oEyeDir = normalize(oWorldPos - eyePosition);
061 }
062
063 //... Sun Functions
064
065 // Fragment program for distorting a texture using a 3D noise texture
066 void main_fp(
067         float2 noiseCoord           : TEXCOORD0,
068         float4 projectionCoord      : TEXCOORD1,
069         float3 eyeDir               : TEXCOORD2,
070         float3 pos                  : TEXCOORD3,
071         out float4 col      : COLOR,
```

```
072         uniform float4 tintColour,
073         uniform float noiseScale,
074         uniform float3 g_vEyePt,
075         uniform float3 g_vSunColor,
076         uniform float3 wind,
077         uniform float  foamSpread,
078         uniform float  foamHmax,
079         uniform float  foamHmin,
080         uniform sampler2D fresnelMap : register(s1),
081         uniform sampler2D normalMap : register(s2),
082         uniform sampler2D reflectMap : register(s3),
083         uniform sampler2D refractMap : register(s4),
084         uniform sampler2D foamMap : register(s5),
085         uniform sampler2D perlinNormalMap : register(s6)
086         )
087 {
088
089     float4 Color;
090
091     float2 final = projectionCoord.xy / projectionCoord.w;
092
093     // Normal
094     float3 normal = tex2D(normalMap, noiseCoord.xy * noiseScale).rgb;
095
096     eyeDir = normalize(eyeDir);
097     normal = normalize(normal);
098     float3 R = normalize(reflect(eyeDir, normal));
099
100     // Fresnel
101     float2 fresnelPos = float2(eyeDir.x,eyeDir.z);
102     float fresnel = tex2D(fresnelMap, fresnelPos).r;
103
104     fresnel = saturate(fresnel);
105
106     // Reflection / refraction
107     float4 reflectionColour = tex2D(reflectMap, final);
108     float4 refractionColour = tex2D(refractMap, final) + tintColour;
109
110     // Foam
111     wind = normalize(wind);
112     float delta = saturate((pos.y-foamHmin)/(foamHmax-foamHmin));
113     float foam = pow( delta,foamSpread) * saturate(dot(wind,normal));
114
115     float4 foamColor = tex2D(foamMap, noiseCoord.xy * noiseScale * 15);
116
117     // Final colour
118  Color = lerp(refractionColour, reflectionColour, fresnel) +  foamColor *
g_vSunColor * foam;
119
120     // Fog
121     //... Realistic Fog and Sun Calculations
122
123     col = Color;
124
125 }
```

Glossário

A

AABB(*Axis-aligned bounding box* em inglês, ou Caixa alinhada com os eixos cartesianos) é um volume envolvente de um objeto 3D.

Abstract Factory é um padrão de projeto de software. Este padrão permite a criação de famílias de objetos relacionados ou dependentes, através de uma única interface e sem que a classe concreta seja especificada.

API, de *Application Programming Interface* (ou Interface de Programação de Aplicativos) é um conjunto de rotinas e padrões estabelecidos por um software para utilização de suas funcionalidades por programas aplicativos - isto é: programas que não querem se envolver em detalhes da implementação do software, mas apenas usar seus serviços.

B

Bufferé uma região de memória usada para armazenar informações.

C

CPU (*Central Processing Unit* em inglês, ou Unidade Central de Processamento), é a parte de um computador que interpreta e leva as instruções contidas no software.

CSV (*Comma-separated values* em inglês, ou Valores separados por vírgula) é um formato de arquivo que armazena dados tabelados, cujo grande uso data da época dos *mainframes*. Por serem bastante simples, arquivos .csv são comuns em todas as plataformas de computador.

CUDA é uma API de programação da NVIDIA, que utiliza uma linguagem derivada do ANSI C para suporte ao processamento utilizando os recursos da GPU.

D

DLL(*Dynamic-link library*em inglês, ou Biblioteca de ligação dinâmica) é a implementação feita pela Microsoft para o conceito de bibliotecas compartilhadas nos

sistemas operacionais *Microsoft Windows* e OS/2.

F

Far Plane Plano de corte do volume de visão de uma câmera virtual. Consiste na base maior do *Frustum*.

FPS(*Frames per Second* em inglês, ou Quadros por segundo) é a métrica mais usada para teste de desempenho de sistema gráfico. Ela consiste em calcular quantos quadros um sistema/algoritmo é capaz de gerar em 1 segundo.

FFT(*Fast Fourier Transform* em inglês, ou Transformada Rápida de Fourier) é uma ferramenta matemática utilizada para transformar sinais e funções do espaço do tempo para espaço da frequência.

Fragments ou Fragmentos são os possíveis pixels a serem mostrados na tela, sua exibição depende do tratamento feito pelo *Pixel Shader*.

Framework ou arcabouço é uma estrutura de suporte definida em que um outro projeto de software pode ser organizado e desenvolvido. Um *framework* pode incluir programas de suporte, bibliotecas de código, linguagens de script e outros softwares para ajudar a desenvolver e juntar diferentes componentes de um projeto de software.

Frustumé o volume de visão de uma câmera virtual. Sua representação pode ser entendida como o tronco de uma pirâmide.

G

Game Engine, ou Motor de Jogos é um programa de computador e/ou conjunto de bibliotecas, para simplificar e abstrair o desenvolvimento de jogos ou outras aplicações com gráficos em tempo real, para videogames e/ou computadores rodando sistemas operacionais

GPGPU (*General-purpose computing on graphics processing units*), ou também referido como GP²U é uma unidade de processamento gráfico que pode executar código específico, assim como uma CPU.

GUI (no Brasil, interface gráfica do usuário; abreviadamente, a sigla GUI, do inglês *Graphical User Interface*) é um mecanismo de interação homem-computador. Com um mouse ou teclado o usuário é capaz de selecionar esses símbolos e manipulá-los de forma a obter algum resultado prático. Esses signos são designados de *widgets* e são agrupados em kits.

GPU(*Graphics Processing Unit* em inglês, ou Unidade de Processamento Gráficos) é o núcleo de processamento das placas gráficas.

H

HLSL (*High Level Shader Language*) é uma linguagem utilizada pelo *Microsoft DirectX* para programar *vertex* e *pixel shaders*.

I

I/O é uma sigla para Input/Output, em português E/S ou Entrada/Saída. Este termo é utilizado quase que exclusivamente no ramo da computação, indicando entrada de dados por meio de algum código ou programa, para algum outro programa ou hardware, bem como a sua saídaou retorno de dados, como resultado de alguma operação de algum programa.

IFFT (*Inverse Fast Fourier Transform* em inglês, ou Transformada Inversa Rápida de Fourier) é uma ferramenta matemática utilizada para transformar sinais e funções do espaço da freqüência para espaço do tempo.

K

Kernelé um termo com vários significados, contudo, no contexto presente, ele é usado como a definição de uma função central de processamento, ou seja, o núcleo de processamento de dados.

L

LoD (*Level of Detail*) são técnicas que envolvem a redução da complexidade de cenas e objetos 3D a fim de reduzir o consumo de processamento.

Logs são registro de atividades gerado por programas de computador.

M

Memory leak, ou vazamento de memória, é um fenômeno que ocorre em sistemas computacionais quando uma porção de memória, alocada para uma determinada operação, não é liberada quando não é mais necessária. A ocorrência de vazamentos de memória é quase sempre relacionada a erros de programação e pode levar a falhas no sistema se a memória for completamente consumida.

Microsoft DirectX é uma coleção de APIs que tratam de tarefas relacionadas a programação de jogos para o sistema operacional Microsoft Windows, ou seja, é quem padroniza a comunicação entre software e hardware. O DirectX foi inicialmente distribuído pelos criadores de jogos junto com seus produtos, mas depois foi incluído no Windows.

MipMapsão coleções de imagens pré-calculadas de uma imagem maior, que visam acelerar o processo de desenho e reduzir artefatos visuais.

Motor de jogo(Idem Game Engine).

MVC (*Model-view-controller*) é um padrão de arquitetura de software, nele é realizada a separação entre os dados (*Model*) e o layout (*View*). Desta forma, alterações feitas no *layout* não afetam a manipulação de dados, e estes poderão ser reorganizados sem alterar o *layout*.

N

Near Plane Plano de projeção do volume de visão de uma câmera virtual. Consiste na base menor do *Frustum*.

O

OpenGL (*Open Graphics Library*) é uma especificação definindo uma API multiplataforma e multi-linguagem para a escrita de aplicações capazes de produzir gráficos computacionais 3D (bem como gráficos computacionais 2D).

P

Pixel (*PictureElement*, ou seja, elemento de imagem) é o menor elemento num dispositivo de exibição ao qual é possivel atribuir-se uma cor. De uma forma mais

simples, um *pixel* é o menor ponto que forma uma imagem digital.

Pixel Shader é um*shader* que manipula pixels por meio de efeitos aplicados a cada um deles na tela.

Plotar é desenhar (uma imagem, especialmente um gráfico) baseando-se em informação fornecida como uma série de coordenadas.

R

Rasterizar é a operação de converter uma imagem vetorial em uma imagem *bitmap*.

Renderização é o processo pelo qual se podem obter imagens digitais. Este processo aplica-se essencialmente em programas de modelagem e animação (3ds Max, Maya, entre outros), como forma de visualizar a imagem final do projeto bidimensional ou tridimensional.

S

Shader na área da computação gráfica significa um conjunto de instruções de software, que serão utilizadas em uma GPGPU para a produção de efeitos de renderização.

SIMDé um método de operação de computadores com várias unidades operacionais em computação paralela. Neste modo, a mesma instrução é aplicada simultaneamente a diversos dados para produzir mais resultados. O modelo SIMD é adequado para o tratamento, cuja estrutura é muito regular, como as matrizes e vetores.

Stream é um fluxo de dados. Quando um arquivo é carregado para ser editado, esta carga ocorre num fluxo, ou seja, linha a linha até o carregamento total do arquivo, como água a correr num cano ou bytes sendo lidos por um programa.

T

Toolkit é um conjunto de *widgets*, elementos básicos de uma GUI. Normalmente são implementados como uma biblioteca de rotinas ou uma plataforma para aplicativos que auxiliam numa tarefa.

U

U.D.é um acrônimo de unidades de distância.

V

Vertex shader é um *shader* capaz de trabalhar na estrutura de vértices do modelo 3D.

W

Widget é um termo sem tradução que designa componentes de interface gráfica com o usuário (GUI). Qualquer item de uma interface gráfica é chamada de *widget*, por exemplo: janelas, botões, menus e itens de menus, ícones, barras de rolagem, etc.

X

XML (*eXtensible Markup Language*) é uma recomendação da W3C para gerar linguagens de marcação para necessidades especiais. Ela é capaz de descrever diversos tipos de dados. Seu propósito principal é a facilidade de compartilhamento de informações através da Internet.